TELECOMMUNICATIONS NETWORKING AT THE START OF THE 21ST CENTURY

Technologies and Building Blocks for Fast Packet Forwarding

Werner Bux, Wolfgang E. Denzel, Ton Engbersen, Andreas Herkersdorf, and Ronald P. Luijten
IBM Research

ABSTRACT

We provide a review of the state of the art and the future of packet processing and switching. The industry's response to the need for wire-speed packet processing devices whose function can be rapidly adapted to continuously changing standards and customer requirements is the concept of special programmable network processors. We discuss the prerequisites of processing tens to hundreds of millions of packets per second and indicate ways to achieve scalability through parallel packet processing. Tomorrow's switch fabrics, which will provide node-internal connectivity between the input and output ports of a router or switch, will have to sustain terabit-per-second throughput. After reviewing fundamental switching concepts, we discuss architectural and design issues that must be addressed to allow the evolution of packet switch fabrics to terabit-per-second throughput performance.

INTRODUCTION

At the turn of the century, the world is facing a fascinating phenomenon: the establishment of the Internet as a worldwide communications medium for the entire spectrum of communication modes: data, voice, video — both real-time and non-real time. The Internet's growing popularity for entirely new applications in the fields of e-business and entertainment as well as its growing use for well-established applications such as telephony have resulted in a spectacular annual growth factor (4–10) of traffic carried by the net.

The fact that the Internet is able to grow at this enormous pace is — apart from economic factors — enabled by the multiplication of optical transmission bandwidth made possible by wavelength-division multiplexing (WDM) and commensurate progress in the packet forwarding capability of the network nodes (routers and switches).

Figure 1 shows the anatomy of a modern router or switch with its main functional units: *line interfaces*, which physically attach multiple transmission systems to the node and provide framing functionality; *network processors*, which provide the intelligence and processing power to analyze packet headers, look up routing tables, classify packets based on their destination and source addresses and other control information and (often complex) rules, and provide queuing and policing of packets; the *switch fabric*, which provides high-speed (ideally nonblocking) interconnection of the node's packet processing units; and the *system processor*, which performs control point functions such as route computation and box and network management. In this article we focus on the two critical functions involved in the *forwarding of packets*: packet processing and box-internal switching. We then describe the evolution of requirements and technical solutions, and discuss techniques that promise to provide the functionality, performance, scalability, and flexibility required in tomorrow's routers and switches.

Since the introduction of optical fibers in transport networks, the serial time-division multiplexing (TDM) — synchronous optical network/synchronous digital hierarchy (SONET/SDH) transmission speed has grown exponentially at a rate of about 30 percent/year to reach 40 Gb/s today (Fig. 2a). The speed increase is primarily gated by the electronics of the transceivers, which suggests that the data rates will level off in the not too distant future. Despite amazing progress in high-speed semiconductor technologies, it is difficult to imagine today that serial transmission rates of commercial transmission systems will grow much higher than 100 Gb/s because of the intrinsic complexity and resulting costs of the transceiver electronics, and, most important, the availability of a much cheaper alternative to increase fiber transmission utilization in the form of WDM. Deployment of WDM transmission technology brought about a radical change: suddenly, the overall transmission capacity of fiber links grew at a rate of about 200 percent/year, already reaching 1.6 Tb/s (160 x 10 Gb/s or 40 x 40 Gb/s). WDM technology is deployed pervasively in the core transport networks and is about to emerge in metropolitan networks. Although the WDM capacity trend is expected to continue for some time, longer-term physical limits will cause saturation — probably on the order of about 50 Tb/s.

0163-6804/01/$10.00 © 2001 IEEE

The port speeds of switches and routers inevitably had to follow the speed increase of serial transmission over fibers (Fig. 2b). This was made possible by advances in complementary metal oxide semiconductor (CMOS) technology combined with design optimizations in packet processing and switching hardware. The proliferation of WDM transmission systems has given rise to an interesting question: are routers and switches going to deal with the multiplication of fiber transmission capacity by a corresponding increase of port speeds? For reasons that will become clear in the discussion below, we are convinced that port speed will continue to increase in line with fiber serial transmission rates, but that the necessary growth in packet-forwarding capacity will come from the nodes' increase in size rather than port speed (Fig. 2c).

Building bigger systems implies two things: distributing the packet processing over more processing units, and making switches with many more input and output ports than available in today's designs (Fig. 2d). The technical challenges resulting from these new requirements are discussed below.

Network Processors

Today's network nodes typically employ application-specific integrated circuits (ASICs) to achieve packet forwarding and classification performance commensurate with the data rates of the attached links (so-called *wire-speed* performance). Such standard ASICs have a typical development cycle after specification of 12–18 months, and, although fast in terms of processing and economical in terms of silicon area and power consumption, are rarely flexible enough for rapid adaptation to protocol or standards changes.

A new type of device promises to solve this problem. Instead of having special ASICs designed for each switch, router, or WAN access device, communications equipment manufacturers will implement the performance-critical packet forwarding functions in software that execute on special-purpose *network processors* (NPs). Thus, manufacturers will be able to add, expand, or modify functions for layer 3–7 packet processing by modifying the NP software instead of making time-consuming and expensive hardware changes. Dataquest predicts that the programmable communications processor market will reach $1 billion by 2003, which explains the amazing investments startup companies and established communications technology vendors are making in the development of NP technology [1].

To illustrate the operation principle of an NP, Fig. 3 shows a generic block diagram. Through the "to and from PHY/switch fabric" interface, data of multiple physical interfaces or the switch fabric are transferred to/from the NP. The bitstream processors receive the serial stream of packet data and extract the information needed to process the packet, such as the IP source/destination address, type of service (TOS) bits, or TCP source/destination port numbers. The packet is then written into the packet buffer memory. The extracted control information is fed to the processor complex, which constitutes the programmable unit of the NP. Under program control, the processor, if needed, extracts additional information from the packet and submits the relevant part to the search engine, which looks up the medium

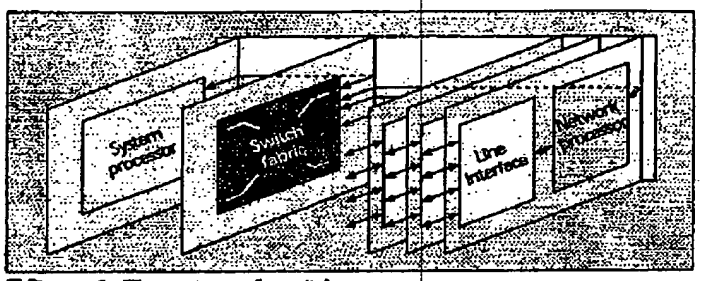


Figure 1. *The anatomy of a switch or router.*

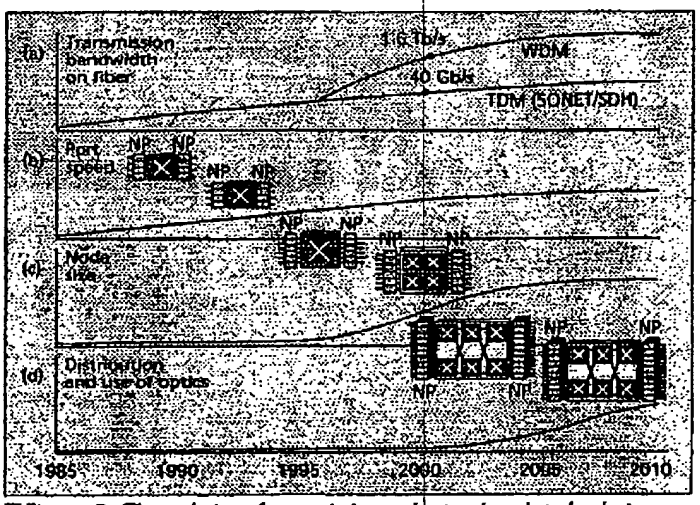


Figure 2. *The evolution of transmission and network node technologies.*

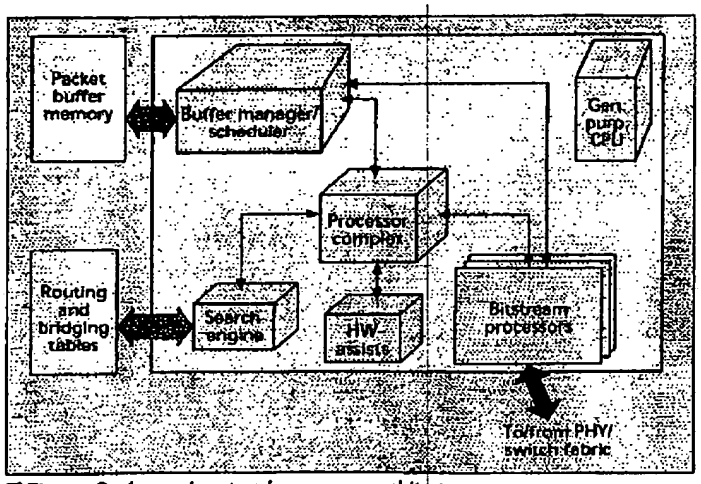


Figure 3. *A generic network processor architecture.*

A key technical challenge is to ensure that the interprocessor communication overhead (to preserve the packet sequence and the synchronization of data flow-related state information) does not cancel the performance gain of parallel processing.

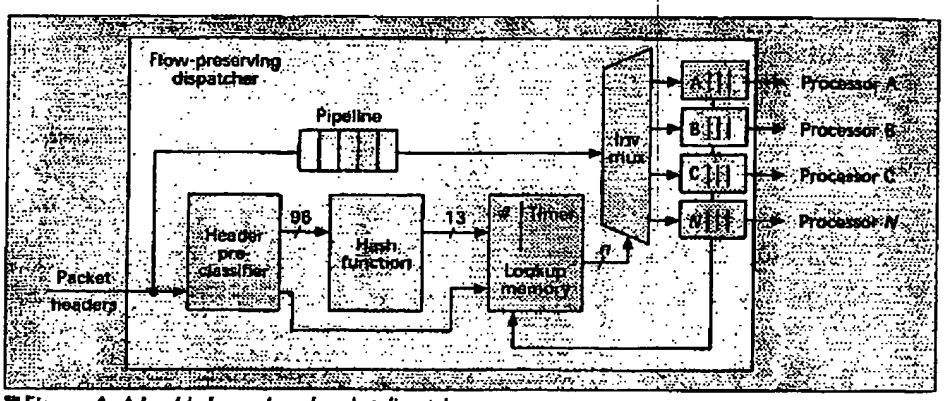


Figure 4. *A load-balancer-based packet dispatcher.*

access control (MAC) or IP address, classifies the packet, or does a virtual circuit/path identifier (VCI/VPI) lookup if the packet is recognized as an asynchronous transfer mode (ATM) cell using the routing and bridging tables and appropriately designed hardware assists. Based on the results returned, the processor instructs the scheduler to determine the appropriate departure time of the packet. Upon packet transmission through the bitstream processor, the necessary modifications to the packet header are performed.

Taking a closer look at the three major building blocks (processor complex, packet buffer memory, and lookup and classification engine), we shall now discuss present and future requirements, and the resulting design issues and implementation challenges.

The Processor Complex

Internet backbone link rates will evolve in the foreseeable future from today's OC-48 (2.5 Gb/s) to OC-192 (10 Gb/s) and even OC-768 (40 Gb/s). With a minimum packet size in the range of 40 bytes (TCP/IP ACK or SYN packets), these rates translate into wire-speed forwarding requirements of 6, 25, and 100 million packets/s.

Application benchmarks on general-purpose CPUs arrive at 2–3 instructions/packet byte for a single routing table lookup [2]. Taking into account that simple layer 2/3 forwarding operations require multiple lookups per packet (MAC address resolution, MAC address learning, IP lookup) a minimum CPU performance of 2.5, 10, or 40 billion instructions/s for handling OC-48, OC-192, or OC-768, respectively, will be required. More advanced networking functions such as virtual private networks (VPN) and quality of service (QoS) typically use encryption, data compression, and packet classification, which requires one to two orders of magnitude more processing power.

Today's NP products cover the performance range up to OC-48 and provide the necessary processor performance with on-chip multiprocessor clusters that employ optimized instruction sets and dedicated hardware assists to offload performance-critical functions: address lookup, classification, encryption/decryption, header checksum calculation, and so on.

These hardware assists usually function as coprocessors, whereby instruction calls are integrated as elementary machine instructions in the instruction set architecture of the CPU. This way, complex functions, which would require a substantial amount of native processor instructions of the CPU, can be dispatched with a single instruction and executed concurrently. A prerequisite for the actual increase in packet throughput is that the CPU does not idle while waiting for a coprocessor to complete its operation. Idling is avoided by either providing multithreading capabilities in the processor via hardware-assisted register-bank swapping to simultaneously work on multiple packets, or by a sophisticated pipeline architecture of the entire NP in which the offload functions complete their tasks prior to using the respective results in the main processor. For both solutions, it is important that the instruction set be open to third-party software providers for code base and tool development.

Although today's high-end general-purpose processors can execute on the order of 1 billion instructions/s, specialized instruction set processors rarely achieve the system clock rates and architectural features of their multi-issue superscalar or very long instruction word (VLIW) counterparts. The hardware assists described above may reduce the required amount of CPU instructions per packet by a factor of 3 (for simple forwarding) to 100 (for more complex encryption/decryption and data compression). However, this does not suffice to bring the workload to a low enough level for a single processor.

For these reasons, today's high-end NPs employ multiple (e.g., 16) multithreaded processor cores clustered into one processor complex. A key technical challenge is to ensure that the interprocessor communication overhead (to preserve the packet sequence and synchronization of data flow-related state information) does not cancel the performance gain of parallel processing. A network node that distorts the packet order may cause an excessive amount of end-to-end retransmission and is therefore unacceptable. One way to tackle this problem is to use an ordering unit. On the arriving side, a dispatcher dynamically assigns packets to a free

processor. Once the processor has finished processing the packet, it indicates this to the ordering unit, and the packet is enqueued in the outbound transmit buffers. The ordering unit is in charge of maintaining packet sequence within a particular packet flow and usually works with the scheduler unit to enforce this. In addition, as each processor is eligible to process packets from any flow, state information must be kept in shared memory, and mechanisms for proper serialization of data access and consistency must be provided.

An alternative scheme for solving the workload assignment problem is shown in Fig. 4. In this scheme, the flow-preserving intelligence resides in the dispatcher, which uses a fixed deterministic function to assign flows to processors: all packets of one flow will be assigned to the same processor. This simultaneously solves the packet sequencing problem and serialization of access to pertinent packet-flow data. In the example shown, a header preclassifier extracts unique flow identification data (IP addresses, TCP port numbers, etc.) from the packet header. A deterministic, static hash function compresses the flow ID to a 13-bit index, which is used to address a moderate-sized SRAM lookup table to map the flow to a specific processor. When a packet pertaining to a new flow arrives (i.e., the SRAM table entry is empty), it is assigned to the processor with the currently lowest load, and the SRAM table entry is updated accordingly. The actual load of the processor is estimated by monitoring the queue lengths of buffers *A*, *B*, ..., *N*. Performance evaluations of this approach using real Internet traffic flows have shown that the queuing buffers, which absorb the difference between aggregate packet throughput of the entire processor complex and the individual processors' capacity, are drastically reduced in size compared to round-robin or other non-feedback-based flow-to-processor assignment algorithms. Hence, the queuing buffers can be implemented on-chip, which enables extremely fast realization of the load-balancing mechanism.

Packet Buffer Memory

As in many other communication subsystems, memory access bandwidth to the external DRAM-based packet data repository is the scarcest resource in NPs. For this reason, the NP's architecture must be designed very carefully to avoid unnecessary data transfer across this memory interface.

In an NP architecture as depicted in Fig. 3, each packet byte may traverse the memory interface up to four times when encryption/decryption or deep packet parsing functions are performed. This is also the case for short packets such as TCP/IP acknowledgments, where the packet header is the entire packet:

- Write packet to data store on inbound
- Read header (= packet) into processor complex
- Write back to memory
- Read for outbound transmission

This means that for small packets, which typically represent 40 percent of all Internet packets, the required memory interface capacities amount to 10, 40, or 120 Gb/s for OC-48, OC-192, or OC-768, respectively. Even the lowest of these values, 10 Gb/s, exceeds the access rate of today's commercial DRAMs. Complex memory-interleaving techniques that pipeline memory access and distribute individual packets over multiple parallel DRAM chips can be applied for 10 Gb/s and possibly 40 Gb/s memory subsystems. At 120 Gb/s, today's 166 MHz DDR SDRAMs would require well over 360-bit-wide memory interfaces, or typically some 25 DDR SDRAM chips. This suggests that at OC-768 speeds, only on-chip, ultra-wide DRAM technology can support the required memory access rates.

The Lookup and Classification Engine

In the mid-1990s, forwarding tables of Internet backbone routers contained some 10,000 entries; today they exceed 85,000, and expectations are that routing tables with up to 500,000 entries may be required in fewer than five years [3]. Emerging security and class-of-service requirements, with their need for packet classification, add a new dimension to the packet forwarding problem: In order to find and apply the appropriate rule from the classification rule base, multiple searches or lookups per packet are required. The challenge is to combine high forwarding and classification performance with low memory usage of classification and forwarding tables. The dynamic policy-based networking of tomorrow's Internet will require a highly dynamic classification rule base that supports table update frequencies on the order of hundreds of updates per second.

Over the past few years, significant progress has been made in the development of forwarding algorithms and implementations. Most techniques, however, address only a subset of the above parameters (speed, size, and update performance) [4]. Only recently have algorithms been proposed that address the classification and forwarding problem in a generic way based on worst-case assumptions and no longer rely on special properties of the forwarding and classification tables and rules [5]. Related work at IBM Research [6] has concentrated on finding approaches that introduce pipelining by segregation of the forwarding key in suitable bit fields and then dynamically deciding whether the longest-prefix matched (LPM) lookup of a field is done as a bit test or a table lookup, depending on whether the forwarding table is locally sparsely filled. This approach tends to hold the information on a specific prefix localized and not compressed, allowing fast updates. Lookup times equal a single memory access cycle and the table size scales better than $O(P)$, with P being the number of prefixes in the forwarding table. Classification can be decomposed in a similar fashion, allowing parallel range searches. The results of the range searches are combined into a variable-sized prefix, which can be resolved through the above LPM lookup to yield the final classification result.

Because of the inherent high degree of parallelism in these approaches, it is expected that within the next few years, packet classification and forwarding implemented in hardware will

Only recently have algorithms been proposed that address the classification and forwarding problem in a generic way based on worst-case assumptions and no longer rely on special properties of the forwarding and classification tables and rules.

Complexity and cost prohibit generously sized output buffers, hence packet losses remain an issue. To overcome this problem, various improvements of the generic OQ concept have been proposed.

achieve OC-768 link speeds for forwarding table sizes of 500,000 entries or more, tens of thousands of classification rules, dynamically updatable in sub-millisecond time, and all stored in on-chip DRAMs.

Outlook: Parallel Network Processors

Although parallel processing techniques are employed inside a single NP (e.g., in the processor complex), no true multi-NP approach, where a set of NPs cooperate to give the appearance of a single higher-performance NP, has been presented yet. Load balancing mechanisms, such as the one described above in the context of packet header dispatching, have the potential to become the key to an efficient multi-NP solution. For example, in order to build an OC-768 NP solution out of lower-speed NPs, one could split the 40 Gb/s data stream into multiple 10 Gb/s or lower-rate streams using the flow-preserving technique described above. Such a solution would dramatically reduce the memory interface problems discussed in the "Packet Buffer Memory" section.

Moreover, this technique could be extended to provide another very attractive feature: since the header preclassifier in the load balancer is aware of the protocol characteristics of the data flows, it may overrule the hash result and assign specific flows (e.g., IPsec or ATM) to processors optimized for handling these flows. Thus, a heterogeneous set of processors could be supported as an alternative to having every processor implement all functions.

Switch Fabrics

Switch fabrics serve to interconnect the various functional units of a switch or router, in particular network and system processors (Fig. 1). The two basic functions of a packet switch fabric are the spatial transfer (switching) of packets from their incoming ports to the destination ports and the buffering of packets to resolve contention. In this section, following a brief review of switch architecture fundamentals, we discuss recent architectural advances in the form of virtual output queuing in combination with either pure input queues or combined input/output queues, and use an implementation of the latter as a case study to point out present and future challenges in the realization of high-speed packet switch fabrics. The section concludes with a look into the more distant future of switching.

Classic Switch Architectures

The two classic single-stage packet switch architectures are characterized by the temporal order of queuing and switching functions [7]. Queuing before switching is called *input queuing* (IQ) (Fig. 5a), and switching before queuing is *output queuing* (OQ) (Fig. 5b). The two architectures have different performance behavior. For uniform Poisson traffic, OQ achieves 100 percent throughput with infinite FIFO output buffers, whereas IQ is limited to 58 percent throughput due to the head-of-line blocking phenomenon. For nonuniform or bursty traffic the efficiency of IQ can be even worse. In both cases, finite buffers may cause packet losses.

The attractiveness of IQ lies in its simplicity and low cost. However, in the early days of fast packet switching, performance was the reason why many switch designs adopted the OQ concept in spite of the more complex and expensive multiport buffers required to enqueue multiple simultaneously arriving packets destined for the same output port (e.g., Bell Labs' Knockout, IBM's Prizma, NEC's Atom, or Siemens' Sigma switch). Complexity and cost prohibit generously sized output buffers; hence, packet losses remain an issue. To overcome this problem, various improvements of the generic OQ concept have been proposed: A first improvement is the shared queuing (SQ) concept [8] (Fig. 5d), which reduces the loss probability from that of dedicated output queues due to better utilization of the limited memory space available on the VLSI chips. A second improvement of OQ or SQ is in combination with IQ [8] (Fig. 5f), which eliminates loss in the output queues by backpressure; that is, the input queues hold back packets if they no longer fit into the output queues.

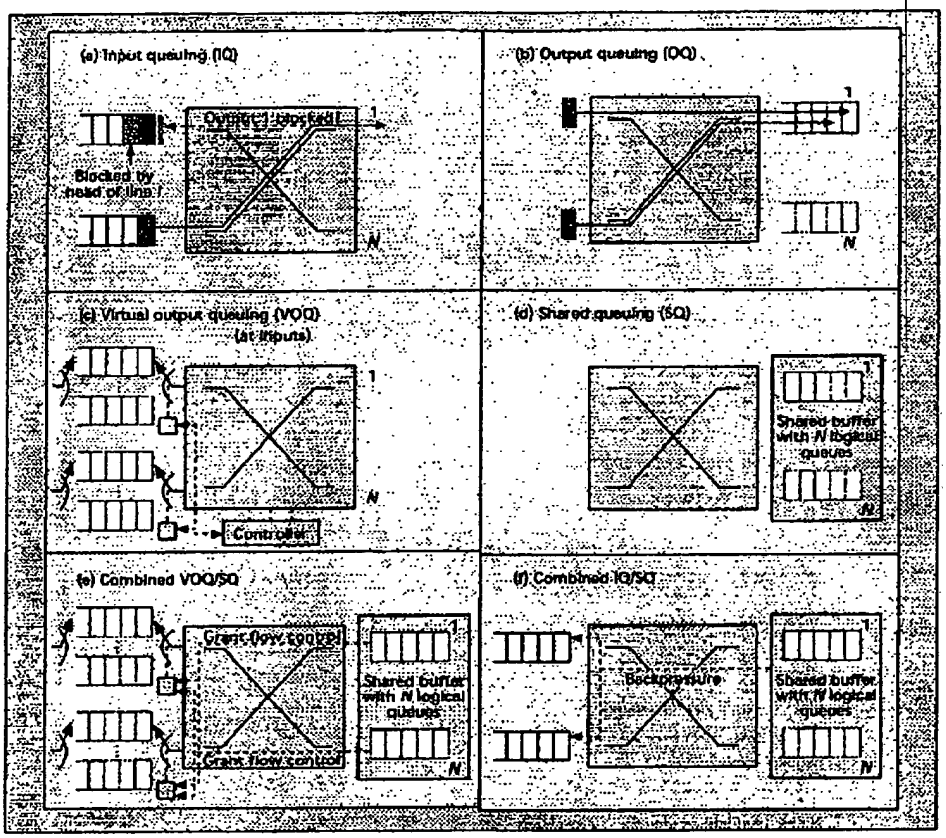
Virtual Output Queuing

It is well known that a more sophisticated queuing discipline can avoid the IQ head-of-line blocking problem. What is needed is to provide a separate queue per output at each input (i.e., a total of N^2 input queues for an $N \times N$ switch), and an appropriate scheduling algorithm for these queues that has global knowledge and hence must be centralized. This concept is called *virtual output queuing* (VOQ) (Fig. 5c), although the queuing physically occurs at the inputs. Initially, VOQ did not receive much attention because of the N^2 complexity and the limited scalability of the centralized controller. However, advances in CMOS technology and algorithmic improvements have recently changed this. The N^2 input queues have become easier to implement, and the scalability of the centralized controller has been simplified to some degree by heuristic, suboptimal, though reasonably well-performing scheduling schemes such as the iSLIP algorithm [9].

An alternative approach combines VOQ with SQ [10] (Fig. 5e). Here, the shared buffer provides a repository for the heads of all input queues and hence serves as a contention resolver. Consequently, only simple decentralized schedulers are required at the input ports, a major advantage of this technique. Compared to the combination of SQ with simple IQ, the throughput performance of this concept is very robust with respect to varying traffic characteristics because head-of-line blocking is eliminated. In the next subsection we describe this concept and key aspects of its implementation in more detail.

PRIZMA, a Switch Combining Virtual Output with Shared Queuing

The PRIZMA architecture [8, 10] is built around an SQ-based switch on a single chip that can be combined with either the IQ or VOQ concepts implemented on switch input adapters (Fig. 5e and f). The $N \times N$ PRIZMA chip is a



■ **Figure 5.** *Switch architectures.*

To provide quality of service (QoS), each output port logically supports four priority queues with a transmission priority scheduling and a guaranteed bandwidth mechanism.

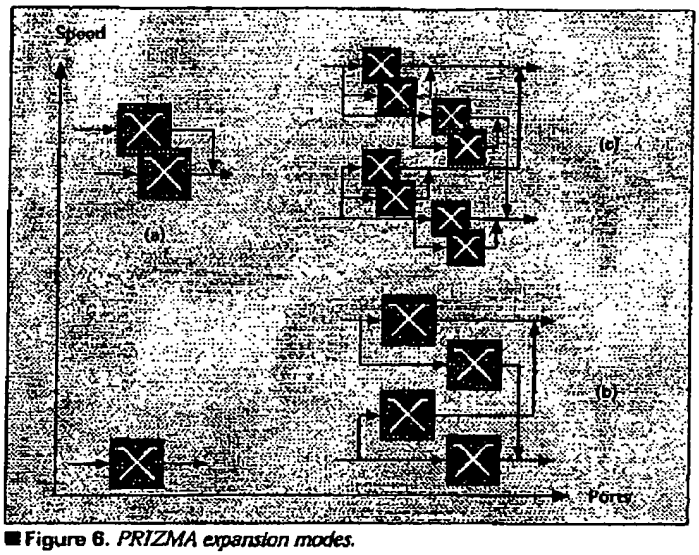
lossless, nonblocking, and self-routing building block, which autonomously forwards fixed-size packets arriving at any of the N input ports to one or more of the N output ports based on switch-internal port addresses. Packets are physically stored in a shared memory and enqueued by writing a pointer into a logical output queue. Multicast is supported by storing one copy of the multicast packet in the shared memory and replicating the pointer in the necessary output queues. PRIZMA has built-in scaling modes (Fig. 6); particularly important is the ability to multiply the port speed by using multiple switch chips in parallel, which is called *speed expansion* (Fig. 6a), and to accommodate more ports in port expansion mode, by using multiple chips in either a particular single-stage arrangement (Fig. 6b) or a multistage arrangement. Furthermore, speed and port expansion can be combined (Fig. 6c).

PRIZMA's packet length is configurable between 64 and 80 bytes to accommodate 53-byte ATM cells as well as the minimum-size 64-byte Ethernet packet. To provide QoS, each output port logically supports four priority queues with transmission priority scheduling and a guaranteed bandwidth mechanism.

For flow control purposes, each (logical) output queue has a programmable threshold. A grant signal is broadcast to all input adapters for each queue. It is active when the queue occupancy is below its threshold, allowing the input adapter to transmit. Using the grant information, each input adapter can locally implement VOQ in its scheduler.

This architecture combines VOQ, SQ, and grant flow control. It also features distributed, simple schedulers that achieve better scalability than the centralized controller approach (Fig. 5c). Consider, for example, a 32 x 32 port switch at OC-192 speed built in the latter architecture. A 64-byte cell takes 51.4 ns to transmit. During this cell time, the central controller has to schedule 32 queues/input port, a total of 1024 queues. A future 64 x 64 switch with OC-768 port speed would need a single central scheduler that can process 4096 queues in 12.9 ns! In contrast, each of the distributed controllers in the combined VOQ/SQ architecture only has to deal with 32 queues in 51.4 ns for the 32 x 32 example with OC-192 ports. For the 64 x 64 case with OC-768 ports, this grows to only 64 queues in 12.9 ns. Moreover, the VOQ/SQ controllers can employ simple round-robin scheduling that can be implemented in one clock cycle, whereas, even with the optimized iSLIP algorithm [9], the central controller takes more than three clock cycles to converge.

The most recent (second-generation) implementation of the PRIZMA architecture consists



■ Figure 6. *PRIZMA expansion modes.*

of a 32-input and 32-output port switch chip, in which each port runs at 2 Gb/s, achieving an aggregate throughput of 64 Gb/s. The chip is realized in IBM's 0.25-μm CMOS technology. Two chips can be operated in speed expansion mode, which doubles the port speed to 4 Gb/s and the throughput to 128 Gb/s. Currently the third-generation PRIZMA chip is being developed. Employing speed expansion with this chip results in a single-stage fabric of 2 Tb/s; using twofold port expansion in addition will yield 4 Tb/s.

The main implementation challenges of a PRIZMA-type terabit-per-second switch fabric are chip-level wiring, moving extremely high data rates on and off the switch chips, and power dissipation. To illustrate these problems, let us consider a 32 × 32 shared-memory design with 1024 fixed packet memory locations and a port speed of 16 Gb/s. Using 0.1-μm CMOS technology, the clock speed would be 2 ns, resulting in a 32-bit-wide bus for each port-to-memory connection. Each of the 32 input ports must reach any of the 1024 memory locations, which in turn must reach any of 32 output ports. In front of each memory there is a 32-to-1 multiplexer, each input being a 32-bit bus. If we were to draw a fictitious line in front of the 1024 memories, a total of 1024 x 32 x 32 = 1 million wires would cross that line. Even when using an aggressive metal pitch and a high number of metal layers, this fictitious line would have to be 30 cm long, which illustrates the magnitude of the wiring problem to be solved.

The technical problems we face in implementing the required on- and off-chip data rates are equally challenging. If, under the same assumptions as above, we postulate, for chip-to-chip interconnect, a 2 Gb/s serial link technology employing differential signaling and requiring 200 mW/link, eight serial interfaces per port would be needed. Input and output ports counted together, this results in 1024 pins needed for data transfer alone, excluding clocking, power, ground, and other I/Os! The chip power required would be 51 W — just for getting data on and off chip without taking into account the power needed for the chip's switching function. It is obvious that novel technological and engineering approaches are required to overcome these implementation problems.

Further Down the Road

WDM technology is significantly increasing the number of channels to be switched for the same number of fibers. This means that the pre-WDM port numbers of 8–32 are growing to hundreds, if not thousands, for the core switching nodes. Whereas advances in CMOS technology and single-stage expansion initially allowed the number of ports to grow by factors of 2–4 with a moderate number of chips, hundreds or thousands of ports call for modular multistage interconnection fabrics (Fig. 2d). Terabit-per-second switches and routers employing a multistage fabric will require hundreds of switching chips. High-speed interconnection of chips, boards, and racks, and the associated power and space issues will be among the most challenging problems in the design of such systems.

The common belief is that optical technologies will eventually solve most of these problems. Parallel optical interconnects between the stages of multistage switch fabrics are already employed in the most advanced terabit systems. These are adequate to meet the more demanding distance and speed requirements of these large-scale systems without electromagnetic interference (EMI) sensitivity, but they are not yet dense, fast, and cheap enough for the future. The speed per interconnection channel will need to be pushed toward 10 Gb/s. Cost reductions must come from technologies such as improved VCSEL array and VCSEL packaging technology, molded plastic connectors, low-cost optical waveguides embedded in boards, and passive alignment techniques. Ultimately, low-cost (coarse) WDM technology may be used inside boxes to reduce the degree of physical parallelism. Waveguides could be used to realize the necessary multiplexing and demultiplexing functions.

Although it is already clear that optical switches will be deployed in the cross-connect network layer, it is less obvious whether optical switching technologies could partially replace electronic switches in multistage packet switch routers. In particular, once the interstage connections are optical, it would be attractive to realize the center stage(s) of a multistage network by optical switch modules, thereby saving part of the costly opto-electronic and electro-optical conversions of the interconnection and reducing overall power. Since optical switch technologies are too slow to switch on a packet-by-packet basis, this constraint must be compensated for by suitable electronic packet buffering and switching schemes such that longer bursts of multiple packets can be handled in one batch by the optical switch. This resembles concepts invented more than a decade ago, such as fast circuit switching or burst switching. Large-scale switches or routers will continue to be built as hybrid systems

requiring the flexibility of electronic network processors and electronic packet switching modules surrounding a potentially optical switching stage. In the absence of optical memory and logic of sufficient capacity, hybrid switches will be with us for a long time to come.

Concluding Remarks

Bringing the forwarding capabilities of future switches and routers to the level dictated by exploding Internet traffic on one hand and rapidly expanding fiber transmission technology on the other is a task that will challenge the imagination and technical skills of the communications engineering community in the new century. As explained in this article, there are hardware architectures and designs that exploit advances in VLSI and optical technologies, and promise to scale to the necessary data rates and system sizes. These designs need to be complemented by equally scalable and reliable networking software. Tomorrow's NP-based routers and switches will include an amazing number and variety of processors from pico-processors and RISC processors embedded within NPs to one or multiple system processors for node and network control. A system and software structure that optimally distributes both packet-by-packet and control processing tasks among these different processors will be crucial for tomorrow's network nodes. The definition of open NP application programming interfaces such as the CPIX standard is a must for the industry to successfully address this challenge.

Acknowledgments

We are indebted to many colleagues at IBM Research and IBM Microelectronics who, throughout years of joint research and development, helped to create a wealth of networking experience, which we have built on to write this article. Special thanks go to Jan van Lunteren for his valuable input to the discussion of table lookup and classification issues.

References

[1] L. Gwennap, "Net Processor Makers Race Toward 10-Gbit/s Goal," *EE Times*, June 19, 2000, http://eet.com/story/OEG20000619S0011

[2] T. Wolf and J. Turner, "Design Issues for High Performance Active Routers," *Proc. 2000 Int'l Zurich Sem. Broadband Commun.*, Zurich, Switzerland, Feb. 16–17, 2000, pp. 199–205.

[3] http://www.telstra.net/ops/bgptable.html

[4] H. Hong-Yi Zeng and T. Przygienda, "On Fast Address-lookup Algorithms," *IEEE JSAC*, vol. 17, no. 6, June 1999, pp. 1067–82.

[5] P. Gupta and N. McKeown, "Dynamic Algorithms with Worst-case Performance for Packet Classification," *Proc. IFIP Networking*, Paris, France, May 2000.

[6] J. Van Lunteren, "Scalable Address Lookup for IPv4 and IPv6," IBM Research rep., in preparation.

[7] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Trans. Commun.*, vol. COM-35, Dec. 1987, pp. 1347–56.

[8] W. E. Denzel, A. P. J. Engbersen, and I. Iliadis, "A Flexible Shared-Buffer Switch for ATM at Gb/s Rates," *Comp. Networks and ISDN Sys.*, vol. 27, no. 4, Jan. 1995, pp. 611–24.

[9] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Net.*, vol. 7, no. 2, April 1999, pp. 188–201.

[10] C. Minkenberg, T. Engbersen, and M. Colmant, "A Robust Switch Architecture for Bursty Traffic," *Proc. 2000 Int'l Zurich Sem. Broadband Commun.*, Zurich, Switzerland, Feb. 15-17, 2000, pp. 207–14.

Biographies

WERNER BUX [F] (wbu@zurich.ibm.com) holds a Ph.D. in electrical engineering from the University of Stuttgart, Germany. He has been with the IBM Zurich Research Laboratory since 1979, where he currently heads the Communication Systems department. Projects in his area of responsibility include network processing and switching technologies, transport technologies, network access technologies, signal processing, and networking software. He is the present Editor-in-Chief of *Performance Evaluation* and a senior editor of *IEEE Journal on Selected Areas in Communications*. He was co-recipient of the 1981 Stephen O. Rice Prize Paper Award of the IEEE Communications Society in the field of communication theory, and received the Third Millennium Medal from the IEEE in 2000. He has also received two IBM Outstanding Innovation Awards in addition to an IBM Corporate Award for his contributions to the IBM Token Ring Network.

WOLFGANG E. DENZEL earned his M.S. and Ph.D. degrees in electrical engineering from Stuttgart University, Germany, in 1979 and 1986. In 1985 he joined the IBM Zurich Research Laboratory, Rüschlikon, Switzerland, where he was involved in the early development of IBM's PRIZMA concept. He also worked on corporate ATM-based and optical networks, participated in several European projects, and coordinated the ACTS COBNET project. Currently, his interest lies in the impact of optics on the evolution of switching technology.

TON ENGBERSEN has been employed at the IBM Zurich Research Laboratory since 1984. He was instrumental in bringing VLSI design skills to the laboratory in the mid-1980s, and in the early 1990s developed the PRIZMA switch architecture. PRIZMA has become a family of communication switch offerings IBM is marketing through its IBM Microelectronics Division. In 1996–1997 he spent two years at the IBM T. J. Watson Research Center, Yorktown Heights, New York, where he led the initial development of MPLS. Since 1997 he has been managing the Network Technology Research Group at the Zurich Research Laboratory. His current research interests are in networking technology, scaleable network processors, scalable switching technology, and SDH/SONET.

ANDREAS HERKERSDORF received a Dipl.-Ing. degree in electrical engineering from the Technical University of Munich, Germany, in 1987, and a Ph.D. in electrical engineering from the Swiss Federal Institute of Technology (ETH), Zurich, in 1991. Since 1988 he has been with the IBM Zurich Research Laboratory, Switzerland, where he currently manages the Network Processor Hardware group. His areas of interest are high-speed communication networks and systems, and VLSI design methodologies.

RONALD P. LUIJTEN received his Master of Electronic Engineering degree from the University of Technology, Eindhoven, The Netherlands, in 1984. In the same year he joined IBM's Zurich Research Laboratory. He was one of the key architects of the Tree Switch, the predecessor of the PRIZMA switch chip, and in 1993 took on the design of the PRIZMA companion gigabit scalable adapter, which formed the basis of the VLSI ATM chipset. In 1997 he became manager of the PRIZMA team, and currently manages the third-generation PRIZMA-GTI project in conjunction with the high-density/high-speed interconnect project.

The definition of open NP application programming interfaces such as the CPIX standard is a must for the industry to successfully address this challenge.